Customizing the Software Process to Support Avionics Systems Enhancements

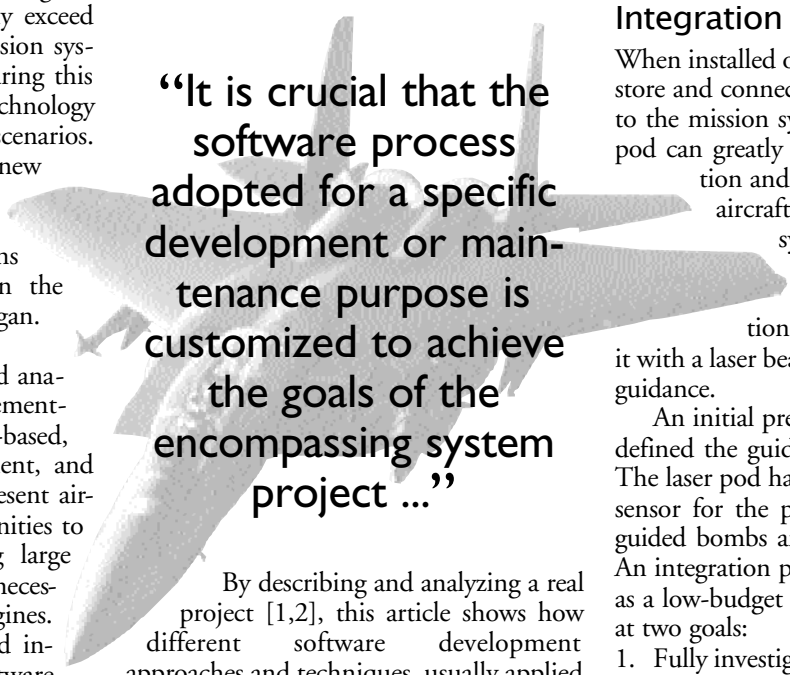
Dr. Paolo Donzelli and Roberto Marozza

When an organization considers enhancing a software-intensive system, particularly an avionics system, selecting the process to be adopted must include consideration of the particular encompassing project. This includes the application domain, the size and complexity of the final product, the hosting system characteristics, etc. Simultaneously, it must be driven by the specific organization's goals, environment, and maturity. By describing and analyzing a real project, this article shows how different approaches and techniques, usually applied in isolation, can be selected, customized and combined to design a software process that better satisfies the organization's goals and meets its constraints. The project was undertaken to investigate the feasibility of enhancing an aircraft mission system by integrating new capabilities, and eventually to identify a quick, low-cost and low-risk solution. An uncertainty-driven product architectural framework together with an ad-hoc simulation-based supporting environment were used to combine in an effective and controlled fashion the waterfall, the Rapid Application Development, and the Incremental Development models.

It is widely accepted that the key point in keeping modern combat aircraft up-to-date is the mission system, i.e., the collection of computers, electronics equipment, and sensors that allow an aircraft to perform its mission. Aircraft operational lifespan has been steadily increasing during the past 60 years, and can now easily exceed three decades (see Figure 1). Mission systems quickly become obsolete during this time due to rapidly advancing technology and fast-changing international scenarios. For example, the capability of new computer systems, or the flexibility required by recent peacekeeping and peace-enforcing operations were not even foreseeable when the design of some modern aircraft began.

As final users, Air Forces are the best candidates to identify and analyze new requirements to be implemented. The availability of standard-based, off-the-shelf sensors and equipment, and the importance of software in present airborne systems offer new opportunities to upgrade aircraft without having large industrial facilities such as those necessary to modify airframes or engines. Thus, many users have developed in-house facilities to evaluate software running on their aircraft, experiment with potential enhancements, and eventually introduce approved modifications.

Since airborne software is vital, yet only a component of a higher complex system, a systemic approach must be adopted in designing the corresponding development process. It is crucial that the software process adopted for a specific development or maintenance purpose is customized to achieve the goals of the encompassing system project, and ultimately of the organization. For military organizations in particular, technical problems posed by the complexity of the target systems often have to be dealt with in combination with specific needs and constraints such as ad-hoc capabilities at time of international crisis, available experience, staff turnover, budget reductions, relations with and between industrial partners, etc.



"It is crucial that the software process adopted for a specific development or maintenance purpose is customized to achieve the goals of the encompassing system project ..."

By describing and analyzing a real project [1,2], this article shows how different software development approaches and techniques, usually applied in isolation, can be selected, customized, and combined to better meet organizational needs. The project was undertaken to investigate the feasibility of enhancing an aircraft mission system by integrating a laser designation pod. Here we describe the software aspects of the integration by focusing on the software process devised to support the integration study, and on the rationale behind the choices made. More details about this specific project can be found in [1,2].

In this article we briefly introduce the integration problem then provide an overview of the proposed software process, and of its rationale. Then we discuss the final results by providing both qualitative and quantitative insights. Finally, we conclude and summarize the benefits of the adopted approach.

Integration Problem Overview

When installed on an aircraft as an external store and connected as an added-in facility to the mission system, a laser illuminating pod can greatly improve both the navigation and attack performance of the aircraft. In practice, such a subsystem will allow the pilot to sight a ground point, obtain the relative position, and eventually illuminate it with a laser beam for subsequent weapon guidance.

An initial pre-feasibility study only had defined the guidelines for the integration: The laser pod had to be used as a targeting sensor for the precision delivery of laser-guided bombs and as a navigation sensor. An integration project was therefore set off as a low-budget short-term activity aiming at two goals:

1. Fully investigate the feasibility of equipping the aircraft with such a subsystem.
2. If feasible, identify an economical and low-risk integration solution.

After an initial assessment of the integration problem, various risk-prone areas were identified.

Complexity of Solution Space

Finding a solution to the integration problem means to define a complete and consistent set of requirements that the new system (i.e., the aircraft equipped with the laser pod) has to satisfy. Only a minor set of these requirements concern functional

aspects, e.g., the data the laser pod has to provide as navigation sensor; most of them are related to non-functional aspects. These include both human factors such as pilot workload, pilot performance, and situation awareness [3], and system quality attributes such as usability, safety, reliability, time, and cost [4].

In comparison with functional requirements, non-functional requirements are highly subjective (e.g., test and front-line pilots can have a different perception of the same problem), strictly related to the particular context, and more difficult to discover, state, and validate without interacting with the final system. This increases the dimension of the solution space, introduces instability in the requirements, and makes it difficult to compare different alternatives.

Complexity of Target Platform

Most of the functions in the mission system are performed via a cooperation of two or more subsystems. As a consequence, modifying or enhancing such functions requires operating on different equipment, which may adopt different hardware and software solutions requiring a broad range of skills not usually available in the same personnel. Moreover, equipment is produced and maintained by different contractors, so that the Air Force is faced with different levels of product visibility, procurement processes, schedules, and costs.

Novel Aspects of Project

Being a single-seat aircraft, the problem of adding the control of the laser pod to all the other tasks usually carried out by the pilot needed to be completely investigated, both for safety and performance reasons.

Project Organization

Based on the initial goals, constraints, and outcomes of the initial assessment of the integration problem (Integration Problem Overview), the project has been organized following some simple guidelines:

- Minimize the impact of integration on the aircraft mission system.
- Exploit internal resources and capabilities, both in terms of personnel and equipment.
- Reuse previous experiences, i.e., lessons learned and available products, e.g., requirements, algorithms, software.
- Allow uncertainty to be part of the project to improve the ability to investigate the solution space.

In practical terms, this has led to making precise choices in organizing the team and defining the software process.

The Team Organization

The project team was structured into three different sub-groups with specific competencies, responsibilities, and workload:

- The software development team, tasked with managing the whole project and developing the necessary software, was composed of two software engineers and four technicians. The group members worked together at the requirements level; subgroups were identified to better deal with the specific needs of the various subsystems affected by the integration.
- The hardware support team, tasked with implementing the avionics integration of the laser pod, was composed of two technicians. In addition, they played the role of logistics and maintenance experts during the requirements definition stage.
- The user group, tasked with collaborating during the requirements discovery and validation phases, was composed of two test pilots. Both of them had a specific experience with laser pod-based operations and were supported by front-line pilots.

Such a team structure provided us with great versatility in tackling personnel-related problems. First, it allowed us to really involve the stakeholders and exploit available expertise. Second, it limited the impacts of staff turnover, unavoidably due to the project schedule, by dealing with them within each group. Third, it incremented the degree of concurrency between the different tasks to be performed within the organization, regarding this or other projects. In particular, both the hardware support team and the user group could better plan their own involvement in this project without hindering other projects. Meanwhile, the software development team, thanks also to the adopted software process (see The Software Process below), was not affected too much by some unavoidable delays that the other subgroups had in providing feedback and support.

The Software Process

The difficulty in adopting a classical waterfall-based process [5], i.e., a process based upon a series of sequential steps that goes from requirements analysis to system delivery, emerged clearly just after a first attempt at defining the initial set of requirements. The waterfall model makes the development process more visible by providing a set of milestones around which the management can plan, monitor, and control a project. However, it is based on the assumption that most of the requirements can be frozen

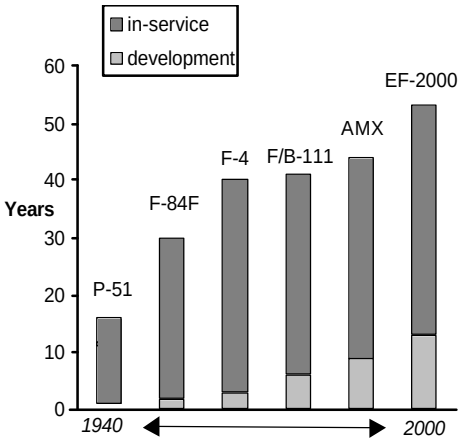


Figure 1: *Aircraft Operational Life*

at the outset of the project, whereas it is well recognized to perform poorly in case of requirements instability [6].

In our case, not only was it impossible to state precisely most of the requirements, but there also were quite a few system areas about which only general and soft considerations could be made. For example, we could identify the need for a pilot interface suitable to reduce workload and improve situation awareness, but we could not translate this into a series of requirements able to be implemented.

Thus a specific architectural framework in which to develop the system was proposed. It had to be flexible to accommodate uncertain software development approaches, yet rigid enough to guarantee a visible development process required by the organization. As illustrated in Figure 2 (see page 12), the software system was designed as a collection of interacting components, each of them acting as a focus for a particular requirements area with a specific uncertainty level.

A set of man-machine interface components, a pod-control component, and a network interface component form the architecture.

- The man-machine interface components represent the software to be added onto the displays (e.g., the head-up display) and the control panels of the aircraft cockpit. They modify the pilot-aircraft interface handling all the information to be displayed and the user interactions enabling the pilot to use the laser pod. Here the level of uncertainty was highest: The user group drafted some guidelines about the pilot operations (mainly on the basis of previous experience), but could not be more precise about the interaction with the individual pilot. Even the specific displays and switches to be used were matters for discussion, leaving the number of components to be developed an open issue.

- The pod-control component implements the algorithms to physically control the pod, for example allowing the pilot to steer the pod in a specific direction. Although many of such algorithms were reused (from available literature, code, etc.), the component still presented some degree of uncertainty, mainly regarding its real-time performance in the new avionics environment. In designing and allocating the software, both measurable parameters (computation elapsed time) and pilot judgement (stability of the track image) had to be taken into account.
- Finally, the network interface component was specifically designed to handle the message passing between the laser pod and the physical systems hosting the other components. This resulted in the most stability, in fact, the laser pod, an off-the-shelf item, dictated most of the messages and the component had to be allocated onto the main digital computer.

The adopted architectural framework (see Figure 2) allowed us to combine different development approaches in a controlled fashion.

A full rapid application development (RAD) approach was applied to the man-machine interface components, which were developed as a set of concurrently evolving prototypes [7]. In RAD, the exact opposite of the traditional software approach is held true: Time and resources are fixed, as far as possible, and the requirements are allowed to change. This suited our situation well; the stakeholders did not have a clear initial idea of the system to be developed. This was expected to mature over time, and different solutions appeared to be equally valid. RAD enabled us to rapidly construct primitive versions of the components by heavily involving the members of the user group.

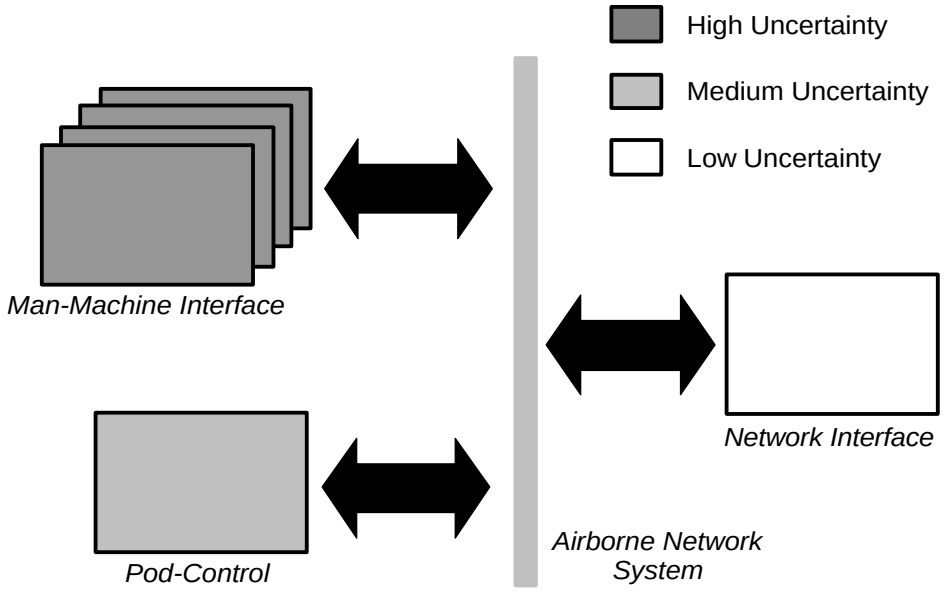
The user group's key role in the development process has been widely recognized. The major advantages of the so-called user-centered approaches [8] come from letting the user participate in and contribute to the design process from its first inception onwards. Through the user group, the users felt that they were an active part of the development team. They took part at the initial design sessions, and their feedback was incorporated to correct, refine, and enrich the emerging system properties until the final set of components was obtained. For the pod-control component, a throw-away prototype [9] was used to quickly compare the different algorithms available before setting off on a more traditional waterfall-based approach. To develop the network interface component, a waterfall-based process was applied from the beginning.

The components were developed in parallel with tight collaboration among the team groups and a continuous exchange of results, experience, and products. An ad-hoc Avionics Simulation Station was developed [10] that supported the whole integration study (see Figure 3). Apart from the Software Development and Testing Stations used to modify the software and test it directly on the airborne computers, the Avionics Simulation Station consists of a mixture of real and simulated equipment. For example, while all the main sensors are simulated (inertial navigation system, radar, etc.), a real laser pod is used together with real units associated with the cockpit (displays, control panels, etc.). The cockpit allows a pilot to form part of the overall station so that the effects of the investigated laser pod integration on pilot performance can be determined.

The Avionics Simulation Station enabled us to work with a variable combination of components running on their allocated computers, and components still at a prototype level. In particular, the network interface component was developed directly on the main digital computer. The other components were implemented using a high-level equipment simulation tool to evaluate them in a real environment without affecting the aircraft equipment. Such a simulation tool allowed us also to adopt for the prototypes the same programming languages used by the potential hosts, highly simplifying the subsequent porting phase.

The prototypes were used to enrich the Avionics Simulation Station with the laser pod. Then the Avionics Simulation Station was employed to analyze the pod operating procedures with the members of the user group. Only when the prototypes reached a stable state, were they moved to the corresponding computers to finalize evaluation and testing. This environment guaranteed the right trade-off between flexibility and rigor to support the integration study. Different alternative solutions were easily investigated (e.g., a different set of components or different allocation of the same components), while the use of real equipment provided early feedback on the project's technical feasibility. For example, the real-time performance of the pod-control component on the selected target computer could be assessed while the user group was still defining the interface to be adopted. Furthermore, it strongly reduced our dependency upon the equipment manufacturers, for example by enabling us to involve them only at the final stage (the porting step), thus reducing the associated costs and delays.

Figure 2: *An Uncertainty-Driven Software Architectural Framework*



Although RAD is a good method to deal with unstable requirements, it suffers when applied to big projects or when requirements instability is not confined to a specific area. In both cases, it can in fact lead to an explosion of project complexity and of associated risks. Whereas the initial architectural framework (see Figure 2) enabled us to manage instability in order to increase our control over the project, we decided to combine RAD with an incremental development method. In other words, the initial unstable set of requirements was broken down into three more manageable subsets:

- Set A regarding the basic laser pod control functions (e.g., in-flight and on-ground pod test, basic pod orientation, etc.).
- Set B incrementing Set A with the laser

pod-based navigation functions (e.g., use of the pod to acquire an on-ground point position).
- Set C incrementing Set B with the laser pod-based attack functions (e.g., use pod capabilities to support a specific ground attack mode).

In moving from Set A to Set C, the complexity of the man-machine interface clearly increased. By requiring more information to be displayed and more controls made available, the incremental approach combined with RAD allowed us to gradually involve the user group, which had time to face new problems (new requirements) starting from a previously established platform (already implemented requirements). Besides facilitating requirements capturing and formalization, such an approach led also to better-designed software, for all the involved components.

The resulting software process is schematized in Figure 4, which summarized what has been said so far. As shown in Figure 4, the process is based on three main phases. Phase 1 is the initial requirement analysis phase during which the decisions of adopting the architectural framework depicted in Figure 2 and of dividing the requirements into incremental subsets were made. During Phase 2, the system evolved as a set of components and prototypes by first implementing the requirements set A (version A), then the set B (version B), and finally the set C (version C). For each version, using Avionics Simulation Station led to correction and refinement loops. We only passed on to Phase 3 when a high confidence in version C was reached. Here the components still at a prototype level were ported on the real equipment, and the final system test and evaluation performed.

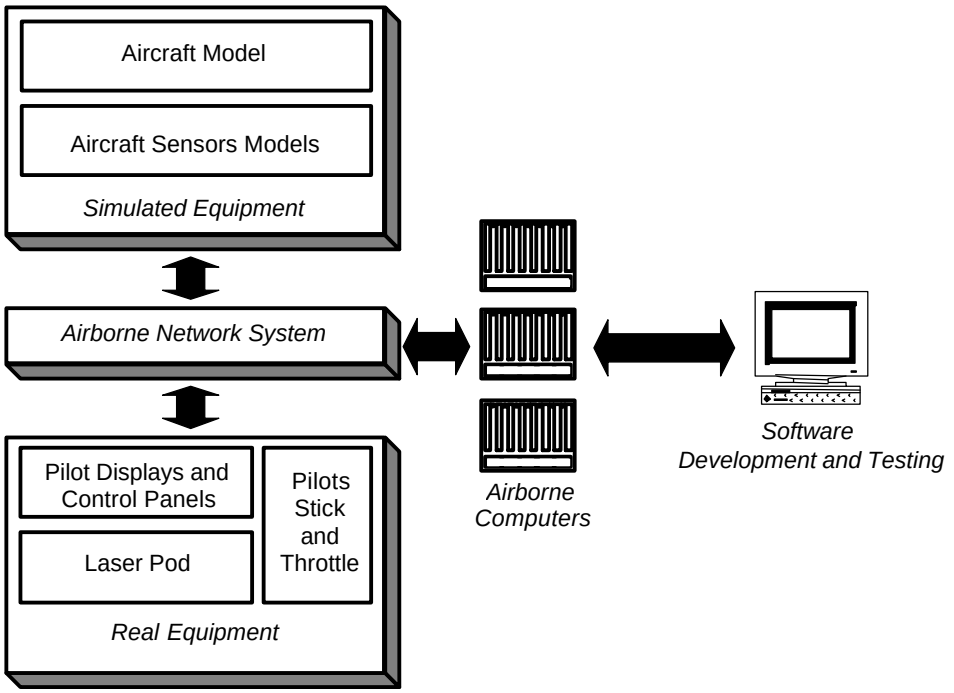


Figure 3: *Architecture of the Avionics Simulation Station*

Project Results

The integration study and the supporting software process allowed us to fully investigate the integration feasibility (goal 1), and then to identify what we considered an economical and low-risk integration solution (goal 2).

The flexibility of the process and product enabled us to find both quantitative and qualitative answers from the early stage of the project. Aspects such as the real-time performance of the modified mission system, the compatibility of the new software with the target equipment, and the pilot needs were carefully investigated. In particular, it was assessed that it was feasible for the laser pod to be operated by the pilot while flying the aircraft.

The software produced consisted of about 11,000 lines of code (LOC); 1,500 LOC of Ada were written as ancillary code to adapt the prototypes to the Avionics Simulation Station. The impact on the total airborne software was relatively small (with an average of 1 percent modified and 5 percent new software on the various airborne computers), thus sensitively reducing the effort required to reevaluate and test the existing functions.

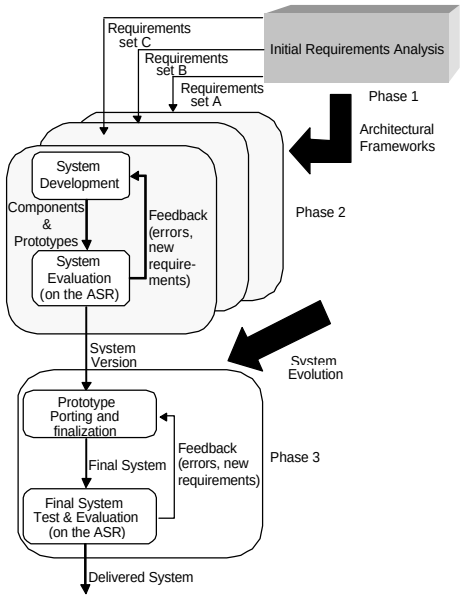
The development lasted for 10 months during a calendar time of 15 months. This difference was due mainly to preemption of personnel for other tasks (about three months) and to some bureaucratic delays with partner industries (about two months). Most of the effort (about 90 percent) was spent on Phase 2, with 45 percent required for development of version A, 30 percent for version B, and 15 percent for version C. Quality confirmation of the components and prototypes forming the system version C, and of the adopted process Phase 3 required only 5 percent of the effort. For the porting, three people from the manufacturers were involved for a limited number of meetings and a total of six days of actual software development. The final testing and evaluation revealed only some minor defects.

Conclusions

This article described the software process adopted to support the integration study of a new weapon system onto an existing military aircraft in the context of a low-cost, high-confidence-of-success project.

To integrate two systems means to identify the synergistic combination that best exploits them both. At the initial stage, only some general guidelines are usually fixed, whereas many different solutions appear equally valid and worth investigating. Such uncertainty drove the design of the initial software architectural framework in which the waterfall, the RAD, and the incremental software development models were combined in an effective and controlled fashion. The process provided the necessary mix of visibility, flexibility, and performance, taking into account the available personnel, time, and funding, as well as increasing the organization's experience and improving collaboration with the industrial partners.◆

Figure 4: *Adopted Software Process*



COMING EVENTS

October 15-18
SEI 16th Annual
Software Engineering Symposium
Washington D.C.
www.sei.cmu.edu/symposium

October 22-26
Systems Engineering
and Supportability Conference
San Diego, CA
http://register.ndia.org/interview/register.ndia

October 28 - November 1
5th Annual DoD Symposium and Exhibition
Kansas City, MO
www.ndia.org

October 29 - November 2
6th Annual Expeditionary
Warfare Conference (EWC)
Panama City, FL
http://register.ndia.org/interview/register.ndia?~brochure~270

November 6-8
TechNet Asia-Pacific 2001
Honolulu, HI
www.afcea.org

November 12-16
5th International Software and Internet
Quality Week - Europe 2001
Brussels, Belgium
www.qualityweek.com

November 13-15
1st Annual CMMI Technology Conference
and User Group
Denver, CO
djenks@ndia.org

February 4-6, 2002
International Conference on COTS-
Based Software Systems (ICCBSS)
At the Heart of the Revolution
Lake Buena Vista, FL
www.iccbss.org

March 19-21, 2002
Federal Office Systems Exposition 2002
Washington D.C.
www.fose.com

April 28 - May 3, 2002
STC 2002
"Forging the Future of Defense
Through Technology"
Salt Lake City, UT
www.stc-online.org

References

1. Donzelli, P., Marozza R. "Laser Designation Pod on the Italian Air Force AMX Aircraft: A Prototype Integration," Proceedings of the NATO/RTO (Research and Technology Organization) Systems Concepts and Integration Panel Joint Symposium on Advances in Vehicle Systems Concepts and Integration, Ankara, Turkey, April 26-28, 1999, published by NATO-/RTO, BP 25, 7 Rue Ancelle, F-92201 Neuilly-Sur-Seine Cedex, France, April 2000, ISBN 92-837-0011-2.
2. Donzelli, P., Marozza R. "Un Laser Pod Anche Per l' AMX," Rivista Aeronautica (Italian Air Force Journal), Dec. 2000, Roma, Italy. <www.aeronautica. difesa.it>.
3. Prince, C., Salas E., and Emery L. "Situation Awareness: What Do We Know Now That the 'Buz' Has Gone?" Engineering Psychology and Cognitive Ergonomics, Vol. 3: Transportation Systems, Medical Ergonomics and Training, pp 215-222. Edited by Don Harris, Ashgate Publishing Ltd, Aldershot, England, 1999.
4. Chung L., and Nixon B. "Dealing with Non-functional Requirements: Three Experimental Studies of a Process Oriented Approach," Proceedings of the International Conference on Software Engineering, Seattle, WA, USA, 1995.
5. Mills, H.D., O'Neill, D., Linger, R.C., Dyer, M., and Quinnan, R.E. "The Management of Software Engineering," IBM System Journal, 24 (2), 1980.
6. Bohem, B. "Anchoring the Software Process," IEEE Software, Vol. 13, No. 14, July 1996.
7. DSDM Consortium. Dynamic Systems Development Method, Version 3. DSDM Consortium, Ashford (UK), 1997.
8. Norman D. and Draper S., "User Centered System Design," LEA, Hillsdale, N.J.,1986.
9. Stytz M.R., Adams T., Garcia B., Sheasby S.M., and Zurita B. "Rapid Prototyping for Distributed Virtual Environment," IEEE Software, Vol. 14 No. 5, Sept./Oct. 1997.
10. Donzelli, P., Moulding M.R. "Developments in Application Domain Modeling for the Verification and Validation of Synthetic Environments: A Formal Requirements Engineering Framework," Proceedings of the Spring '99 Simulation Interoperability Workshop, Orlando, FL, March 1999.

About the Authors

Paolo Donzelli, Ph.D., is an advisor with the Department of Informatics of the Office of the Prime Minister in Italy. A former serving engineering officer with the Operational Testing Centre of the Italian Air Force, Dr. Donzelli was a senior research fellow with the Computing Information Systems Engineering Group, at RMCS, Cranfield University (UK). Dr. Donzelli has a variety of interests in the software engineering area, and his Ph.D. thesis was in software process quality modeling.

Office of the Prime Minister
Department of Informatics
Via della Stamperia 8
00187 Roma, Italy
Phone: (39) 0335-736-5194
Fax: (39) 06-6779-4736
E-mail: p.donzelli@palazzochigi.it

Roberto Marozza was a serving engineering officer with the Italian Air Force before moving into industry. He has been program manager of the on-board mission software for the Anglo-Italian EH-101 ASW helicopter, and has also worked in some airborne software projects for the Tornado and the AMX attack aircraft. Recently, he was involved at Aerospatiale in the software requirements definition of the ATV, the orbiting vehicle that will transfer unmanned payloads from ground to the International Space Station. Marozza has a Laurea Degree in electronics engineering.

Banca d'Italia
Largo Guido Carli 1
Frascati, 00040 Roma, Italy
Phone: (39) 0348-654-2067
E-mail: rmarozza@libero.it